Impact of the execution context on Grid job performances

Tristan Glatard[1,2], Diane Lingrand[1], Johan Montagnat[1], Michel Riveill[1]
[1] Rainbow, I3S joint research laboratory of CNRS and the University of Nice-Sophia Antipolis, France
[2] Asclepios, INRIA Sophia Antipolis, France
{glatard, lingrand, johan, riveill}@i3s.unice.fr
http://rainbow.i3s.unice.fr

Abstract

In this paper, we examine how the execution context of grid jobs can help to refine submission strategies on a production grid. On this kind of infrastructure, the latency highly impacts performances. We present experiments that quantify the dependencies between the grid latency and both internal and external context parameters on the EGEE grid infrastructure. We show how job submission managers, job execution sites and the submission date can be statistically correlated to grid performances.

1. Objectives

Grids are increasingly used as support infrastructures for different scientific application areas [11, 9, 5]. Several grids, such as the European EGEE grid infrastructure[1] [10] or the Open Science Grid (OSG)[2], have reached a production level quality of service. These large scale and multi-users systems are characterized by a non-stationary load, their heterogeneity and their large geographic expansion. As a consequence, non-negligible jobs submission latencies, measured as the time between the jobs submission and the start of their execution, are observed. They are mainly due to queuing systems, network delays and system faults. This system variability is known to highly impact application performances and thus has to be taken into account [14].

Several initiatives aim at modeling grid infrastructure Workload Management Systems (WMS). In [12], correlations between job execution characteristics (job size or number of processors requested, job runtime and memory used) are studied on a multi-cluster supercomputer in order to build models of workloads, enabling comparative study on system design and scheduling strategies. Feitelson [4] has observed correlations between runtime and job size, number of cluster and time of the day.

Models of the grid latency enable the optimization of job submission parameters such as jobs granularity or the timeout value needed to make the WMS robust against system faults and outliers. Properly modeling a large scale infrastructure is a challenging problem given its heterogeneity and its dynamic behavior. In a previous work, we adopted a probabilistic approach [6] which proved to improve application performances while decreasing the load applied on the grid middleware by optimizing jobs granularities. Furthermore, in [7], we show how the distribution of the grid latency impacts the choice of a timeout value for the jobs. An optimal timeout value can be obtained by minimizing the expectation of the job execution time J which can be expressed as follows, where F_R denotes the cdf of the latency, f_R the corresponding pdf, t_∞ the timeout value and ρ the proportion of outlier jobs:

$$E_J(t_\infty) = \frac{1}{F_R(t_\infty)} \int_0^{t_\infty} u f_R(u) du + \frac{t_\infty}{(1-\rho)F_R(t_\infty)} - t_\infty.$$

The resulting timeout value is highly dependent on the nature and parameters of the distribution of R. In particular, the weight of the tail of the distribution of the latency is a discriminatory parameter: heavy-tailed distribution always lead to a finite optimal timeout value whereas for light-tailed ones, it may be better not to set any timeout.

These studies show that determining a precise and up-to-date model of the grid latency is a crucial issue to allow the optimization of job submission parameters. However, the underlying model relies on an estimation of the distribution of the grid latency which is currently collected by the user at a global scale from historical data in absence of specific grid monitoring service. The forthcoming estimation of the distribution may be outdated, thus impairing the accuracy of the estimation, in particular in case of rapid variations of the grid status.

Taking into account contextual information has recently been reported to help in estimating single jobs and work-

[1] http://www.eu-egee.org
[2] http://opensciencegrid.org

flows execution time by rescheduling [13]. We aim at refining our grid model with more local and dynamic parameters. Each job can be characterized by its execution context that depends on the grid status and may evolve during the job life-cycle. The context of a job depends both on parameters internal and external to the grid infrastructure. The internal context corresponds to parameters such as the computer(s) involved in the WMS of a specific job. It may not be completely known at the job submission date. The external context is related to parameters such as the day of the week and may have an impact on the load imposed to the grid.

Our final goal is to improve job execution performance on grids. This requires taking into account contextual information and its frequent update. In this paper, we are studying some parameters among the broad range of contextual information that could be envisaged and we discuss their relevance with regard to grid infrastructures.

2. EGEE infrastructure and data collection

Our experiments are based on the EGEE production grid infrastructure. With 35000 CPUs dispatched world-wide in more than 190 computing centers, EGEE represents an interesting case of study as it exhibits highly variable and quickly evolving load patterns that depend on the concurrent activity of up to 1000 users. For the following discussion, we here introduce the main components of the batch-oriented EGEE grid infrastructure, that are diagrammed in figure 1.

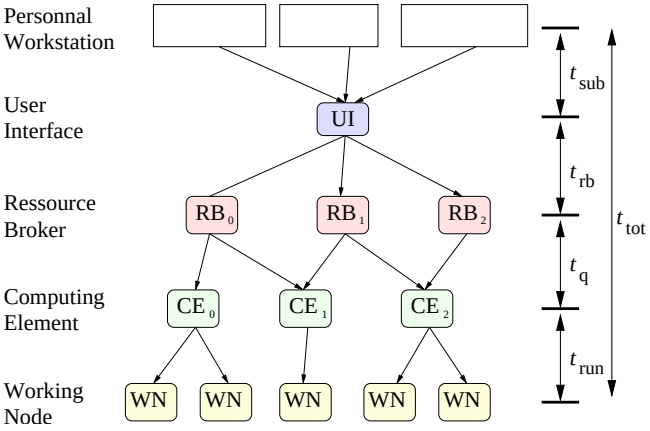

Figure 1. EGEE submission schema.

When a user want to submit a job from her workstation, she connects to an EGEE client known as a User Interface (UI). A Resource Broker (RB) queues the user requests and dispatches them to the different computing centers available. The gateway to each computing center is one or more Computing Element (CE) batch manager that will distribute the workload over the Worker Nodes (WN) in the center. During its life-cycle, a job changes its status. Received by the RB it is initially waiting, then queued at the CE and running on the WN. If everything went right, the job is then completed. Otherwise, it is aborted, timeout or in an error status depending on the type of failure. As shown in figure 1, UIs can connect to different RBs, and RBs may be connected to overlapping sets of CEs.

To study the grid behavior, we have collected measures over the submission of a very large number of probe jobs (*i.e.* jobs only consisting in the execution of a `/bin/hostname` command). We maintained a constant number of probes inside the system by submitting a new one as soon as one completed to avoid introducing any extra variability. The probe jobs were assigned a fixed 10000 seconds timeout beyond which they were canceled and not taken into account (2.5% of the submitted jobs timeouted). The results presented in this document involve the remaining 4477 probe jobs. For each one, we logged the job submission date, the UI used, the UI load at submission time, the RB used, the CE used and the jobs status duration (total duration t_{tot} and partial durations t_{sub}, t_{rb}, t_{q} and t_{run} as illustrated in figure 1).

The cumulative distribution function (cdf) of the latency of this data set is plotted on figure 2. Its median is 363 seconds, its expectation is 559 seconds and its standard deviation is 850 seconds, which quantifies the highly variable behavior of the EGEE grid. The first part of this experimental distribution is close to a log-normal distribution and its tail can be modeled by a Pareto distribution [7]. Pareto distributions are used to model a large class of computer system measurements (jobs durations, size of the files, data transfers length on the Internet...) [8]. This heavy tailed distribution demonstrates that the EGEE grid exhibits non-negligible probabilities for long latencies.

3. Influence of context parameters on the grid latency

Many parameters may have a direct influence on the jobs submitted to a grid infrastructure. We are here focusing on three of them which proved to have a particular impact as shown below: the site of computation (CE), the job dispatcher (RB) used, and the day of the week. The CE and RB are directly related to the specific EGEE grid. However, the results obtained could be extended to other grid by replacing CEs and RBs by the equivalent workload management services. In the DIET middleware [1] for instance, it could correspond to Master Agents (MA) and Local Agents (LA).

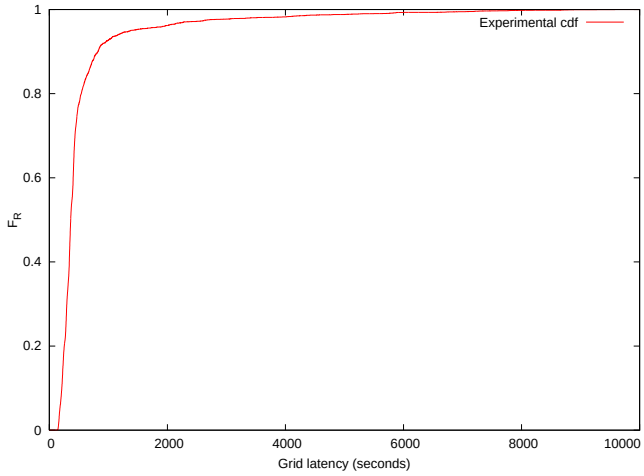

Figure 2. Cumulative density function of the whole experimental data

3.1. Site of computation parameter

The probe measures involved 90 different computing center gateways (CEs) from the infrastructure. Figure 3 plots the cumulative distribution of the grid latency for each CE involved in the experiment. To ensure statistical significance, CEs with less than 30 probe measures were removed from the study. 60 computing elements out of the 90 were remaining.

Figure 3 suggests that 3 classes can be identified among the CEs. A k-means classification was thus done on the cumulative density functions of the CEs and the obtained classes are identified with distinct colors on the figure. Centroids of the classes are plotted in black.

The first class of CEs, pictured in blue, has the highest performance in average. The median of its centroid is 237 seconds. It is composed of 15 CEs. The second class of CEs, pictured in green, is composed of 35 CEs. The median of its centroid is 373 seconds, which corresponds to a 1.6 ratio with respect to the fastest class. Finally, the slowest class, pictured in red, is composed of 10 CEs and the median of its centroid is 652 seconds. Table 1 compares the median, expectation and standard-deviation of the grid latency for each CE class. It reveals that even if the first (blue) class of CEs has the highest performance in average, it is also more variable than the second (green) class. The third (red) class is the most variable. The impact of variability on the performances of an application depends on the number of submitted jobs and on the performance metric. In some cases (high number of jobs), it would be better to submit jobs on a less variable CE class, even if it has the lowest performance in average.

A noticeable feature of the green class is that almost all of its CEs contain the `lcgpbs` string in their names. In this class, the only CE whose name does not contain this string is plotted in cyan on figure 3 and is close to the border of this class. In the blue class, no CE contains this string in its name and in the slowest class, 7 CEs have this string in their name. This shows that the `lcgpbs` string name is informative in itself although the reasons are not necessarily known (it may correspond to a specific middleware version deployed on some of the CEs in this heterogeneous infrastructure).

The order of magnitude of the grid latency thus appears to be correlated to the execution CE. It is relevant because the CE is directly related to the job queuing time as a CE exactly corresponds to a batch queue. Variations of middleware and system versions may explain the differences observed among the 3 different classes while variations inside a given class may be coming from the load imposed by the users and the performance of CEs host hardware.

However, in general, the execution CE is only known after the job submission, during the scheduling procedure. Thus, this information could only be exploited for parameters that can be updated once the job has been submitted, as for instance the timeout value or the application completion prediction, whereas parameters such as the granularity of the tasks to submit could not benefit from the CE information.

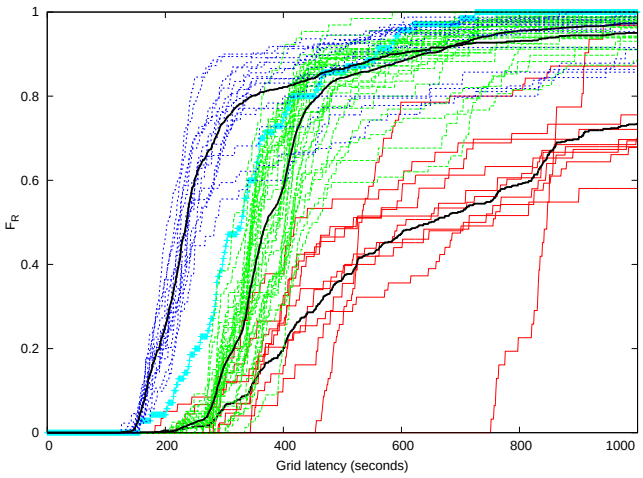

Figure 3. Classification in 3 classes of the cumulative density functions of the grid latencies by CE. Centroids of the k-means classes are plotted in black.

CE group	Median (s)	Expect. (s)	Stdev (s)
not lcgpbs (blue)	237	436	880
lcgpbs (green)	373	461	493
other (red)	652	1132	1396
Whole data	363	559	850

Table 1. First moments and median of the grid latency w.r.t the execution CE class

Resource Broker	Expectation (s)	Stdev (s)
IFCA (red)	19	14
LAL (green)	25	24
SINP (blue)	22	16
Whole data	22	19

Table 2. First moments of the submission time w.r.t the RB

3.2. Resource Broker parameter

The probe measures were submitted to 3 different Resource Brokers (RBs). Figure 4 displays the cumulative density function of the submission time of the probe jobs sent to each of the RBs as well as the one of the submission time considering the whole experimental data set. We first can notice that the submission times seem to be quantified to a discrete set of values that correspond to the ones were the cumulative density function is growing. As every curve correspond to more than 1400 probe measures, this phenomenon does not come from the lack of measures but rather from a characteristic of the submission system. Indeed, to ensure scalability, jobs are sequentially submitted to the RB, which could explain this behavior. The submission time is a multiple of the duration required to submit one job, which is about 4 seconds according to those measures.

The 3 RBs exhibit quite different behaviors. Two of them (red and blue curves) have equivalent tails that are smaller than the one of the third RB (green curve). It indicates that the latter RB is prone to have very high submission delays: on this RB, 30% of the jobs require more than 40 seconds to be submitted, whereas they are less than 10% for the two other RBs. On the other hand, many of the jobs of the green RB are submitted faster than on the two other ones. As a consequence, the median of the submission time on the green RB is only 12 seconds, whereas it is respectively between 19 and 20 seconds and between 15 and 16 seconds on the blue and red RBs.

Table 2 displays the expectation and standard-deviation of the submission time with respect to the RB. Knowing that a job is submitted to the red or blue RB reduces the variance of the submission time distribution. On the contrary, the standard-deviation of the green RB is higher than the one of the whole data.

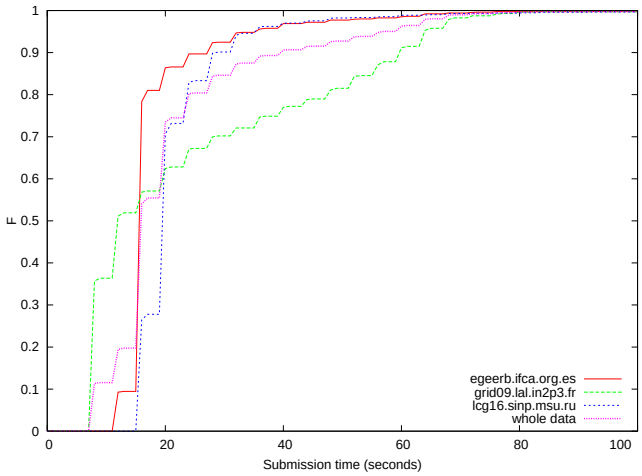

Figure 4. Cumulative density functions of the submission time by Resource Broker

3.3. Day of the week parameter

The day of the week is an important parameter of the external context which is likely to influence the load of the grid infrastructure. Figure 5 plots the cumulative density function of the grid latency with respect to the day of the week and table 3 displays the corresponding expectations and standard-deviations. For this experiment, 1364 probes submitted during the week-end were added to the previous 4477 ones.

The seven days exhibit similar behaviors for latencies lower than 500 seconds. Above this value, Saturday and Sunday have very similar cdf significantly lower than the ones of the other days. In average, those week-end days correspond to the ones when the latency is the highest, as shown by table 3. The variance also seems to be higher during the week-end than during the week.

4. Discussion

We have shown in these experiments that the grid latency is related to the choice of a RB or a CE. More precisely, we have observed that the middleware and system versions are probably involved in this phenomenon. Computers with older systems and middlewares are probably computers that were installed before newer ones, and not upgraded. The differences can thus either come from software performance

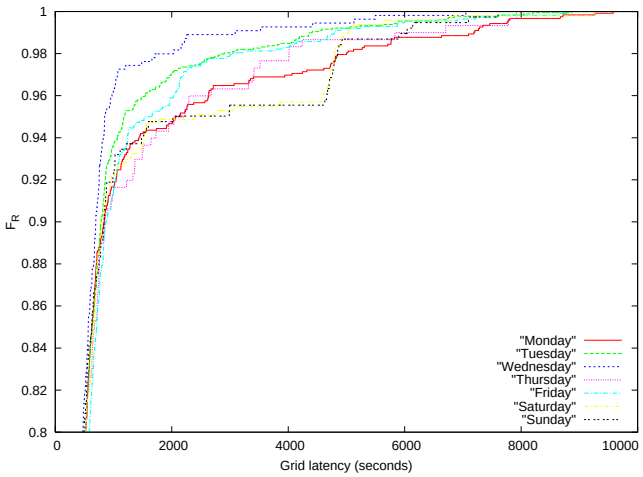

Figure 5. Cumulative density functions of the grid latency for each day of the week

Day	Expectation (s)	Standard-deviation (s)
Monday	622	1088
Tuesday	530	747
Wednesday	461	542
Thursday	609	972
Friday	569	808
Saturday	630	1035
Sunday	629	1066
Whole data	569	886

Table 3. First moments of the grid latency w.r.t the day of the week

improvement or the fact that newer computers have higher computing capabilities. This hypothesis could be confirmed by other experiments establishing what fraction of the latency is due to the computer hardware or to its software. It might be a valuable information for middleware developers. However, from a grid user point of view, the main interest is not necessarily the cause but rather its impact on the applications. A similar approach led Cieslak and co-authors [2] to propose performances analysis through grid log data mining. This can be very efficient in identifying point of failures or performance drops although it usually provide little information on their cause.

The last experiment made shows also an interesting result: days from Monday to Friday are usually accepted as working days while Saturdays and Sundays are usually non-working days. This is the case for most western and eastern countries involved in the EGEE project. However, this assumption fails in some participating countries (for instance, in Algeria the week-end is on Thursday-Friday and in Israel on Friday-Saturday). This information on working days would thus need to be corrected by the geographical location of the grid sites handling the jobs. Working on such a large geographical area also implies to consider the time of the day. Working hours depend on the country we are dealing with and local habits. The dependency between latency and day of the week could be refine considering:

- Local meaning of week-end (*e.g.* Saturday/Sunday or Thursday/Friday).
- Local time of the day (day or night).
- Time zone: days start with significant time shifts in the EGEE infrastructure (from GMT+9 in Japan to GMT-8 in the USA).
- Local habits (*e.g.* working hours).

We believe that the dependency between latency and day of the week is related to the system administrators activity (they are more frequently at work and system or services crashes are more rapidly fixed on week days). However, we also need to consider the fact that there is more activity during the week than during week-end, generating probably more faults. These hypotheses need to be further tested by building a notion of time context with respect to time zones, working days and hours.

Similarly, we can anticipate a correlation between temperature and faults in southern countries: in summer, air conditioning systems cooling down computing centers are more likely to break down, making large amounts of local resources unexpectedly unavailable. CPU's temperature is certainly the most accurate parameter to demonstrate this fact but is often difficult to obtain remotely. Considering cities temperatures could also gives indications on failures probability. This information is easily obtained for large cities through well known Web information providers.

In the future, we plan to examine with more details the execution context. A definition of the context as Dey [3] proposed is probably too general to adapt to our case. For us, context is information that is relevant for our problem. We make a difference between the context of execution we need to study:

- Time context (day, hour, absolute/local, working/not-working, week/week-end).
- Hardware context (CPU, bus, hard disk size and speed, network adapter, bandwidth, load).
- Software (system, middleware).
- Temperature (CPU, city (daily, hourly, ...)).

and the context we will really use for estimating job latencies and optimizing job resubmission strategies. Reaching a very fine granularity (*e.g.* monitoring each computer CPU)

may be intractable in practice and the desired information is not always available. However, higher level information derived statistically such as the CEs and RBs classes may already be sufficient for improving the system performances. Similarly, cities temperature may be both a sufficient and accessible information to further optimize it.

5. Conclusion

We consider the context of execution being composed by the elements that are relevant to our problem. Going into the details of all the parameters of the problem is intractable without a statistical approach. The level of detail we will reach depends both on the availability of the contextual information and the needs of the model.

In this paper, we have exhibited correlations between job submission latencies and parameters from the execution context such as the job dispatcher and batch systems involved in jobs management, or the week of the day. These results encourage to perform more detailed studies in order to have a better understanding of the influence of these parameters.

Depending on their influence and availability, they can be used to refine our model of job latency and to optimize our job resubmission and timeouting strategies. At last, we plan to include our updated model into grid middlewares in the future. In the example of the EGEE grid, there are two levels of modification we will have to consider: at the UI level for choosing the RB and at the RB level for choosing the CE. Furthermore, even if we do not get control on the targets CEs, the timeout value assigned to a job may be dynamically adapted taking into account the destination site processing it.

6. Acknowledgments

This work is partially funded by the French national research agency (ANR), AGIR project[3] (ACI "Masse de données") and NeuroLog project[4] (ANR-07-TLOG-024). This project is in the scope of scientific topics of the STIC-ASIA OnCoMedia project[5]. We are grateful to the EGEE European project for providing the grid infrastructure and user assistance.

References

[1] Eddy Caron and Frédéric Desprez. DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid. *International Journal of High Performance Computing Applications*, 2005.

[2] David Cieslak, Douglas Thain, and Nitesh Chawla. Troubleshooting Distributed Systems via Data Mining. In *IEEE International Symposium on High Performance Distributed Computing (HPDC'06)*, Paris, France, June 2006.

[3] Anind K. Dey. Understanding and Using Context. *Pattern Recognition Letters (PRL)*, 5(1):4–7, 2001.

[4] Dror Feitelson. *Workload modeling for performance evaluation*, pages 114–141. Springer-Verlag - LNCS vol 2459, September 2002.

[5] Tristan Glatard, Johan Montagnat, Diane Lingrand, and Xavier Pennec. Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR. *International Journal of High Performance Computing and Applications (IJHPCA)*, 2007.

[6] Tristan Glatard, Johan Montagnat, and Xavier Pennec. Probabilistic and dynamic optimization of job partitioning on a grid infrastructure. In *14th euromicro conference on Parallel, Distributed and network-based Processing (PDP06)*, pages 231–238, Montbéliard-Sochaux, France, February 2006.

[7] Tristan Glatard, Johan Montagnat, and Xavier Pennec. Optimizing jobs timeouts on clusters and production grids. In *International Symposium on Cluster Computing and the Grid (CCGrid)*, Rio de Janeiro, May 2007. IEEE.

[8] Mor Harchol-Balter. Task Assignment with Unknown Duration. *Journal of the ACM (JACM)*, 49(2):260–288, March 2002.

[9] Nicolas Jacq, Christophe Blanchet, C. Combet, E. Cornillot, L. Duret, K. Kurata, H. Nakamura, T. Silvestre, and Vincent Breton. Grid as a bioinformatic tool. *Journal of Parallel Computing*, 30(9-10):1093–110, 2004.

[10] Erwin Laure, E. Fisher, S.M Fisher, Ákos Frohner, C. Grandi, and Peter Kunszt. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.

[11] Erwin Laure, Heinz Stockinger, and Kurt Stockinger. Performance Engineering in Data Grids. *Concurrency and Computation: Practice & Experience*, 17(2-4), 2005.

[12] Hui Li, David Groep, and Lex Wolters. Workload Characteristics of a Multi-cluster Supercomputer. In *Job Scheduling Strategies for Parallel Processing*, pages 176–193. Springer Verlag, 2004.

[13] Jason Nichols, Haluk Demirkan, and Michael Goul. Autonomic Workflow Execution in the Grid. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(3), May 2006.

[14] Jennifer Schopf and Francine Berman. Stochastic Scheduling. In *Supercomputing (SC'99)*, Portland, USA, 1999.

[3] AGIR: `http:www.aci-agir.org`
[4] Neurolog: `http://neurolog.polytech.unice.fr`
[5] OnCoMedia: `http://www.onco-media.com`